Falsification of OMG infrastructure version 2.4.1

Abstract: We evaluate the OMG infrastructure version 2.4.1 (2011) for: structure and behavior; infrastructure; and superstructure. None is tautologous. Evaluated also are packages named: abstractions; constraints; instances; classes; and constructs. These respectively contain multiple instances of the same named block. The simple example for merging of packages is *not* tautologous. These results form a *non* tautologous fragment of the universal logic VŁ4.

We assume the method and apparatus of Meth8/VŁ4 with Tautology as the designated proof value, **F** as contradiction, N as truthity (non-contingency), and C as falsity (contingency). The 16-valued truth table is row-major and horizontal, or repeating fragments of 128-tables, sometimes with table counts, for more variables. (See ersatz-systems.com.)

LET ~ Not, ¬ ; + Or, ∨, ∪, ⊔ ; - Not Or; & And, ∧, ∩, ⊓, ·, ∘ , ⊗ ; \ Not And;
> Imply, greater than, →, ⇒ , ↦, ≻, ⊃, ↠ ; < Not Imply, less than, ∈, ≺, ⊂, ⊬, ⊭, ↞, ≲ ;
= Equivalent, ≡, :=, ⇔, ↔, ≜, ≈, ≃ ; @ Not Equivalent, ≠, ⊕;
% possibility, for one or some, ∃, ∃!, ◊, M; # necessity, for every or all, ∀, □, L;
(z=z) T as tautology, ⊤, ordinal 3; (z@z) **F** as contradiction, Ø, Null, ⊥ , zero;
(%z>#z) N as non-contingency, Δ, ordinal 1; (%z<#z) C as contingency, ∇, ordinal 2;
~(y < x) (x ≤ y), (x ⊆ y), (x ⊑ y); (A=B) (A~B).
Note for clarity, we usually distribute quantifiers onto each designated variable.

Our approach is to test and validate well known public diagrams from products for IT architecture. Our goal is to find designs as tautologous.

From: OMG unified modeling language (OMG UML). (2011). Infrastructure version 2.4.1.
uml-diagrams.org/uml-25-diagrams.png

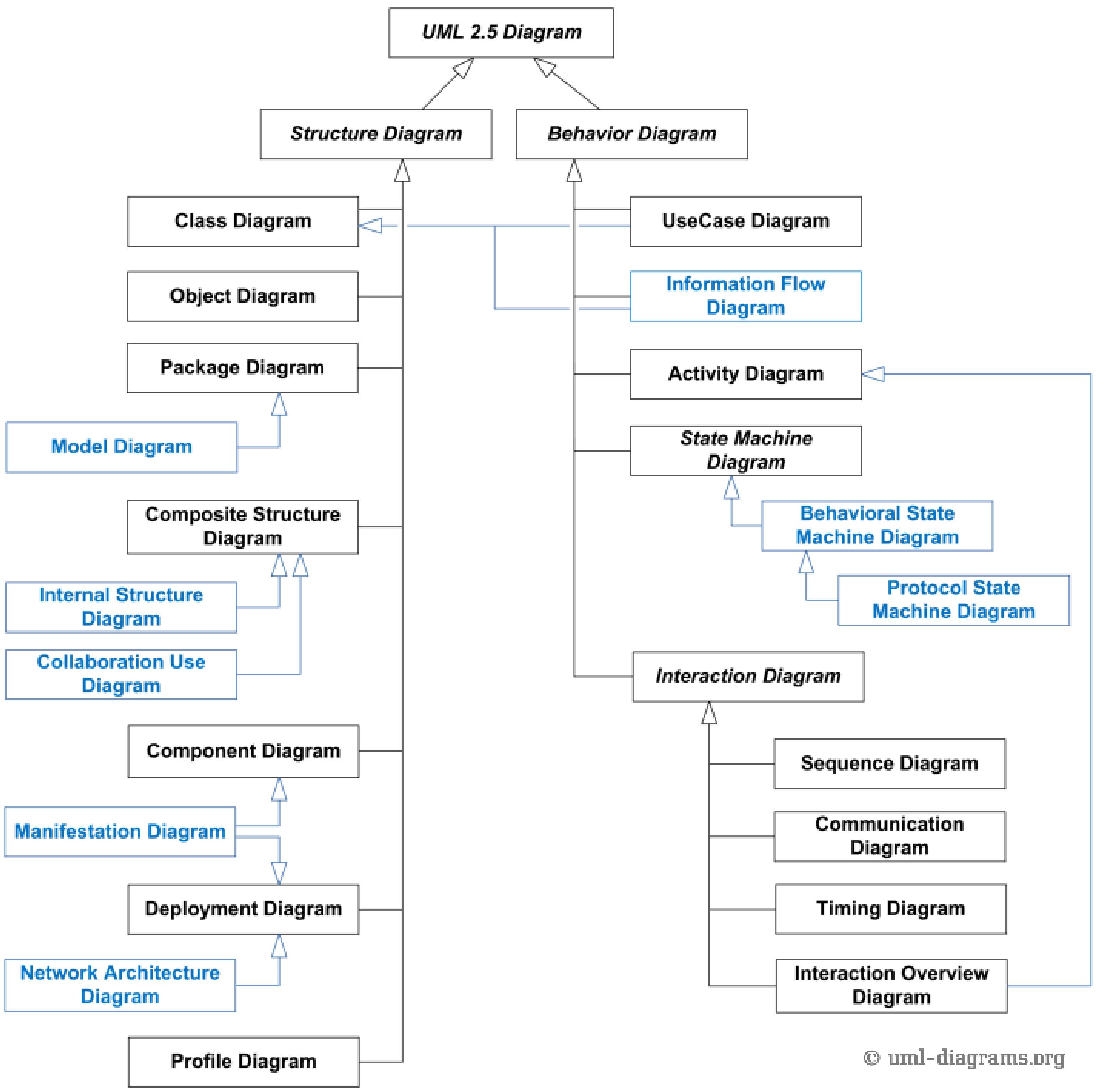
[See next page.]

Left structure_diagram:

LET	p	class_diagram,
	q	object_diagram,
	r	package_diagram,
	s	model_diagram,
	t	composite_structure_diagram,
	u	internal_structure_diagram,
	v	collaboration_use_diagram,
	w	component_diagram,
	x	manifestation_diagram,
	y	deployment_diagram,
	z	network_architecture_diagram.
	[n/a	profile_diagram not used as no supporting block]

Right behavior_diagram:

LET	p
q	use_case_diagram,
r	information_flow_diagram,
s	activity_diagram,
t	interaction_diagram,
u	sequence_diagram,
v	communication_diagram,
w	timing_diagram,
x	interaction_overview_diagram,
y	behavioral_state_machine_diagram,
z	protocol_state_machine_diagram.



(2.5.1.1) (2.5.2.1)

Left behavior diagram: (((p&q)&((r&s)&t)) &((((u&v)&(w&x))>t))&((x>r)&(z>(y>s))) ;

```
FFFF FFFF FFFF FFFF}32}2
FFFF FFFF FFFF FFFT}   }
```

(2.5.1.2)

Right structure diagram: ((((p&q)&(r&t))&(w&y))&((s>r)&((u&r)>t))) &((x>(w&y))&(z>y)) ;

```
FFFF FFFF FFFF FFFF}32  }2
FFFF FFFF FFFF FFFF} 8}2}
FFFF FFFF FFFF FFFF} 4} }
FFFF FFFT FFFF FFFT}  } }
```

(2.5.2.2)

Remark 2.5: The conjunction of Eqs 2.5.1.1 and 2.5.2.1 for UML_2.5_diagram (2.5.3.1) results in:

```
FFFF FFFF FTFT FTFT}16
FFFF FFFF FFFF FTFT}
FFFF FFFF TTTT TTTT}
FFFF FFFF FFFF FTFT}
FFFF FFFF FTFT FTFT}
FFFF FFFF FFFF FTFT}
FFFF FFFF FFTT FFTT}
FFFF FFFF FFFF FFFT}              (2.5.3.2)
```

From: omg.org/spec/UML/2.4.1/Infrastructure/PDF

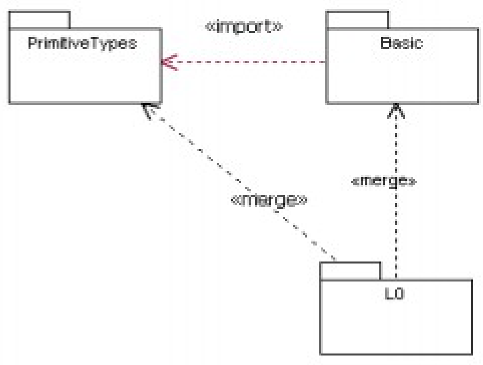


Figure 2.1 - Level 0 package diagram

In this model, "UML" is originally an empty package that simply merges in the contents of the Basic package from the UML Infrastructure. This package, contains elementary concepts such as Class, Package, DataType, Operation, etc.

(2.1.1)

LET p, q, r: primitive_types, basic, LO

((r>q)&(r>q))&(q>p) ;   TT**F**T **FF**TT TT**FT** **FF**TT   (2.1.2)

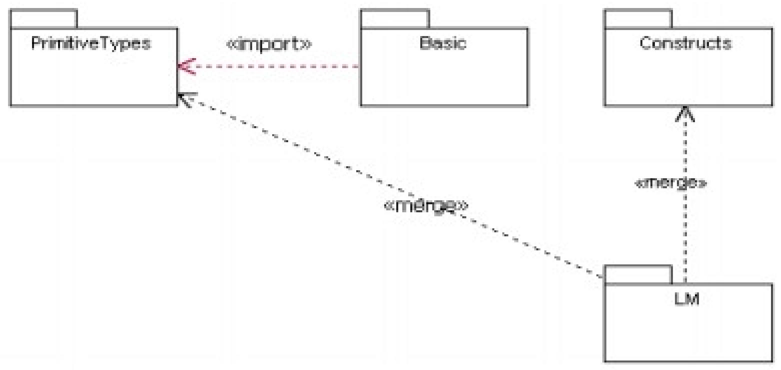


Figure 2.2 - Level M package diagram

Note that LM does not explicitly merge Basic, since the elements in Basic are already incorporated into the corresponding elements in Constructs.

(2.2.1)

LET p, q, r, s: primitive_types, basic, LM, constructs.

((r>q)&(q>p))&(r>s) ;   TT**F**T **FFFF** TT**F**T **FFF**T   (2.2.2)

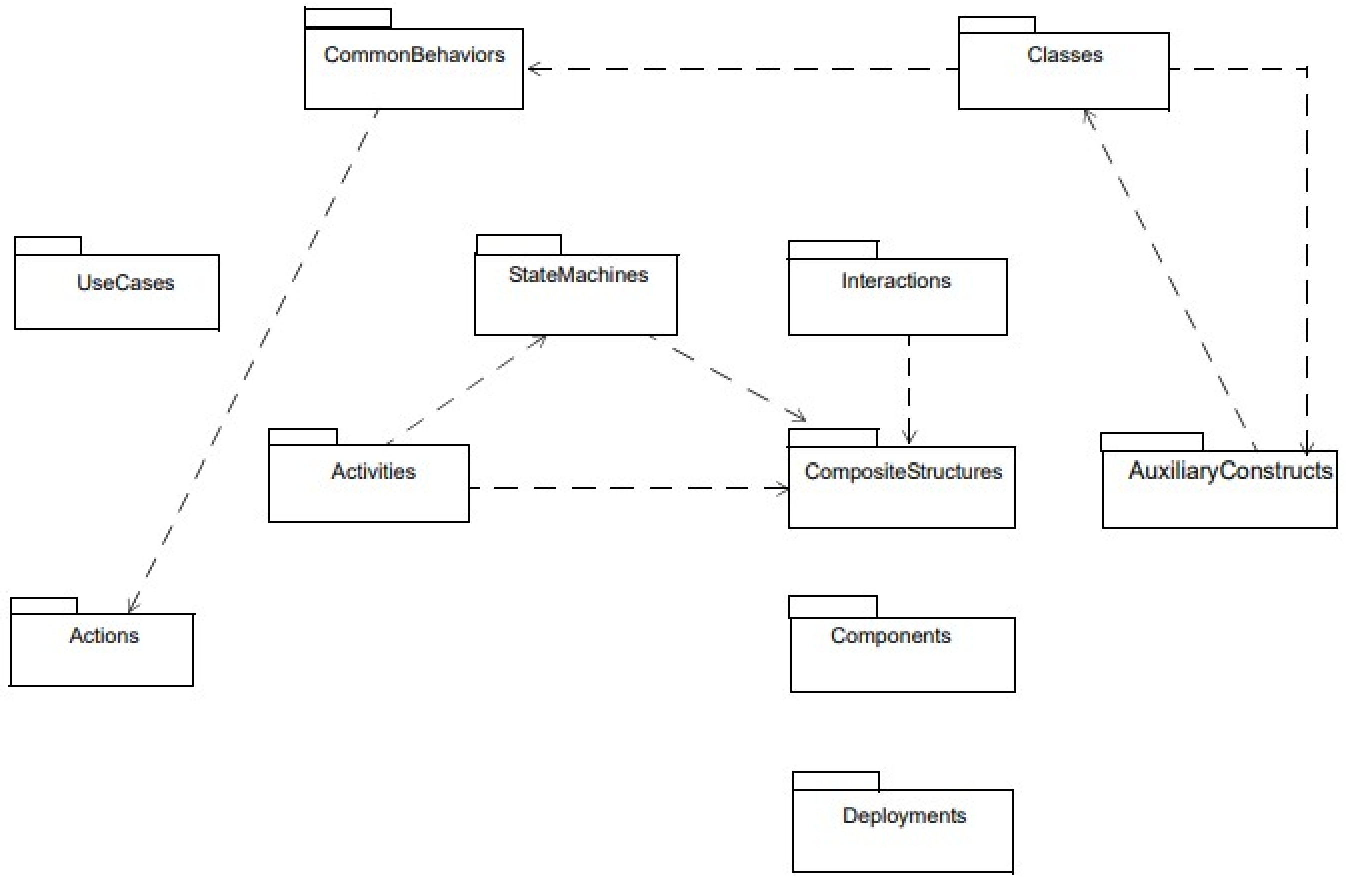


Figure 7.5 - The top-level package structure of the UML 2 Superstructure

(7.5.1)

LET	p	common_behaviors,
	q	classes,
	r	auxiliary_constructs,
	s	actions,
	t	state_machines,
	u	interactions,
	v	activities,
	w	composite_structures,
	x	use_cases,
	y	components,
	z	deployments.

```
(((x&(y&z))=(s@s))&((r>q)>(r&(p>s))))&((u>w)&(v>((t>w)&w))) ;
                                FFFF TTTF FFFF TTTT} 2
                                FFFF FFFF FFFF FFFF} 6
                                FFFF TTTF FFFF TTTT}10}6
                                FFFF FFFF FFFF FFFF} 6}
                                FFFF TTTF FFFF TTTT} 8
                                FFFF FFFF FFFF FFFF}16          (7.5.2)
```

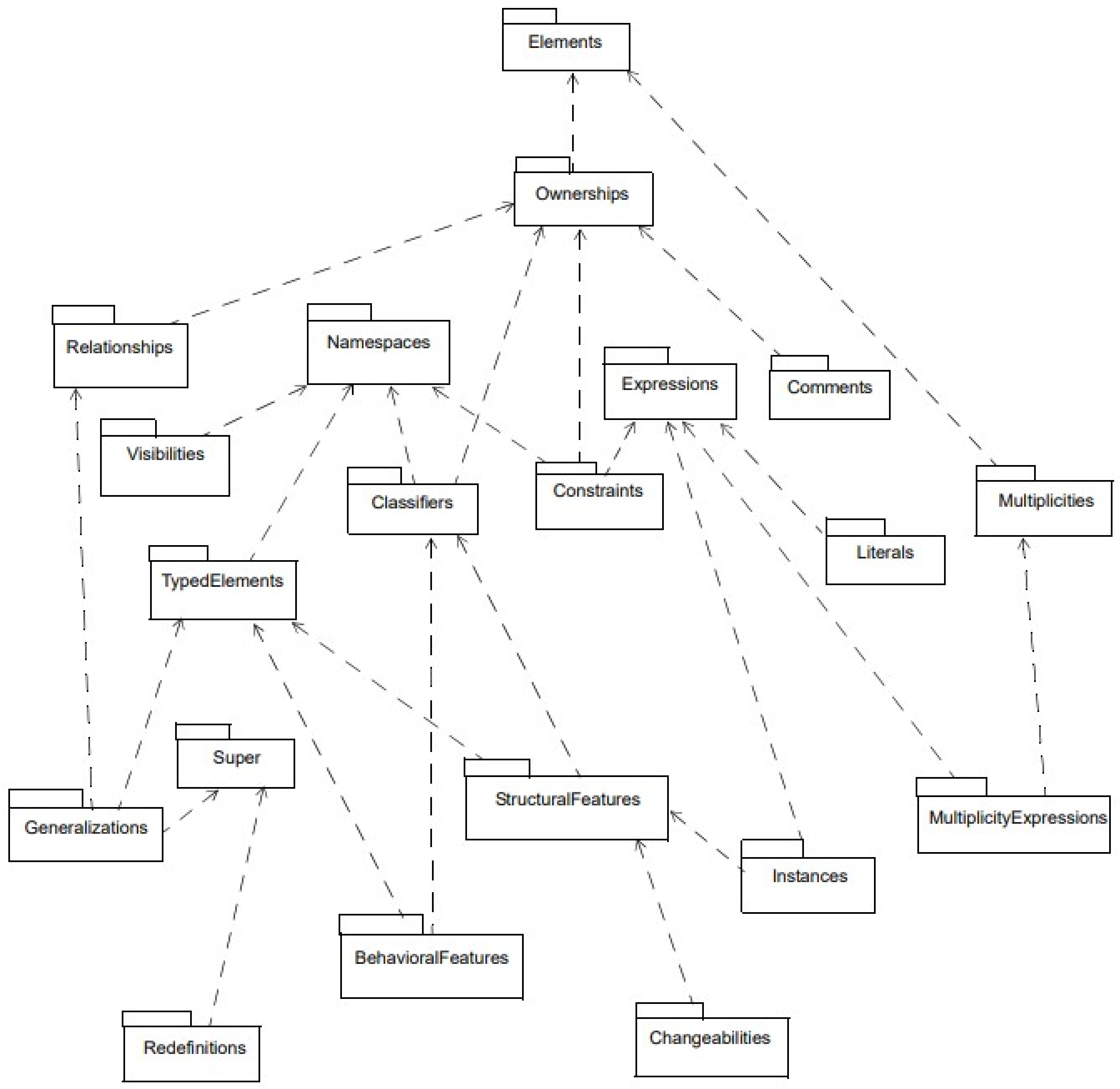


Figure 9.2 - The Abstractions package contains several subpackages, all of which are specified in this clause

(9.2.1)

Remark 9.2.1: Certain blocks are not evaluated, mainly to keep the number of logical propositional variables at 11 or less for timely processing times due to logical complexity. For example, single blocks supporting only one upward block are ignored, such as comments, literals, redefinitions, and changeabilties. Similarly, dead end blocks supporting no upward blocks are ignored, such as name_spaces, expressions, typed_elements, and super.

LET	p	elements,
	q	ownerships,
	r	relationships,
	s	generalizations,
	t	classifiers,
	u	contstraints,

v	multiplicities,
w	structural_features,
x	instances,
y	multiplicity_expressions,
z	behavioral_features.

((((z>t)>(q>p))&((x>w)>t))&(((y>v)>p)&((s>r)>q)))&(u>q) ; (9.2.2)

Remark 9.2.2: Eq. 9.2.2 has the most complex pattern observed in the truth table result. Unique four-row blocks are named as `a`, `b`, `c`, `d`, `e`.

a:
FFFT **FFF**T TT**F**T **FFF**T
FFFT **FFF**T TT**F**T **FFF**T
FFFT **FFF**T **FFF**T **FFF**T
FFFT **FFF**T **FFF**T **FFF**T

b:
FFTT **FF**TT TTTT **FF**TT
FFFT **FFF**T TT**F**T **FFF**T
FFTT **FF**TT **FF**TT **FF**TT
FFFT **FFF**T **FFF**T **FFF**T

c:
FFFT **FFF**T **F**T**F**T **FFF**T
FFFT **FFF**T **F**T**F**T **FFF**T
FFFT **FFF**T **FFF**T **FFF**T
FFFT **FFF**T **FFF**T **FFF**T

d:
FFFF **FFFF** **FFFF** **FFFF**
FFFT **FFF**T TT**F**T **FFF**T
FFFF **FFFF** **FFFF** **FFFF**
FFFT **FFF**T **FFF**T **FFF**T

e:
FFFF **FFFF** **FFFF** **FFFF**
FFFT **FFF**T **F**T**F**T **FFF**T
FFFF **FFFF** **FFFF** **FFFF**
FFFT **FFF**T **FFF**T **FFF**T

The first 64-row block uses `a`, and the second 64-row block uses `b` as follows:

```
e}4
c}2
e}2
e}2
d}
a or b
c
d
e
```

Anomalies for respective UML components were discovered with instances of a same-named block as:

Figure 9.13 - The elements defined in the constraints package
two "element"

Figure 9.25 - The elements defined in the Instances package
three "value_specification"

Figure 10.3 - The classes defined in the Classes diagram
two "type" and three "typed_element"

Figure 11.3 - The Root diagram of the Constructs package
two "comment" and two "element"

Figure 11.16 - The Classifiers diagram of the Constructs package
two "named_element", two "type", and two "typed_element"

Figure 11.21 - The Namespaces diagram of the Constructs package
two "packageable_element" and two "named_element"

Figure 11.25 - The Operations diagram of the Constructs package
two "type"

Figure 11.26 - The Packages diagram of the Constructs package
two "packageable_element"

Component merging:

Examples

In Figure 11.31, packages P and Q are being merged by package R, while package S merges only package Q.

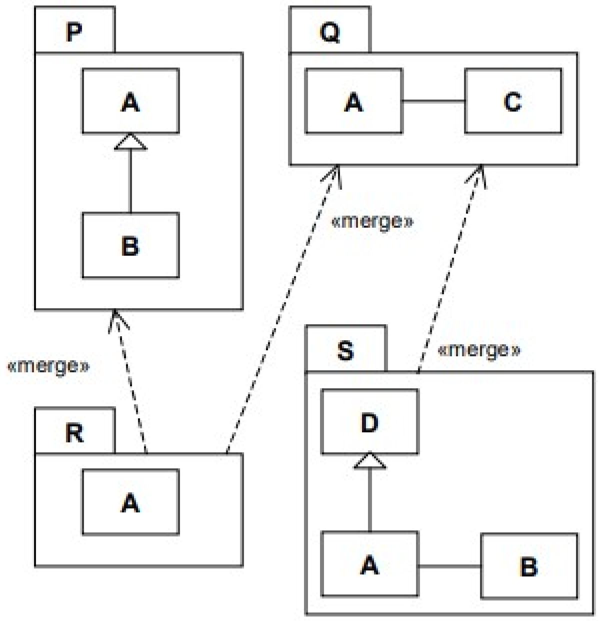


Figure 11.31 - Simple example of package merges

(11.31.1)

LET	p	P,
	q	Q,
	r	R,
	s	S,
	t	A,
	u	B,
	v	C,
	w	D.

(((u>t)&(t>r))>p)&((((t>r)&(t&v))&(((t&u)>w)>s))>q) ;

```
FFFF FFFF FTFT FTFT}8
FFFF FFFF FFFF FTFT}
FFFF FFFF TTTT TTTT}
FFFF FTFT FFFF FTFT}
FFFF FFFF FTFT FTFT}
FFFF FFFF FFFF FTFT}
FFFF FFFF FFTT FFTT}
FFFF FFFT FFFF FFFT}
FFFF FFFF FTFT FTFT}
FFFF FFFF FFFF FTFT}
FFFF FFFF TTTT TTTT}
FFFF FFFF FFFF FTFT}
FFFF FFFF FTFT FTFT}
FFFF FFFF FFFF FTFT}
FFFF FFFF FFTT FFTT}
FFFF FFFF FFFF FFFT}
```
(11.31.2)

Remark 11.31.2: Eq. 11.31.2 as rendered is *not* tautologous.